REMARKS

This responds to the Office Action mailed on October 19, 2006 and the references cited therewith.

Claim 11 is amended; no claims are canceled; and claims 17-24 are added. As a result, claims 1-24 are now pending in this application. No new matter has been added.

Summary of Claimed Subject Matter

As noted in the Background of the present patent application, memory latency is a critical problem with vector processing because of typical use of multiple processing units as well as large amounts of data communication between a vector processor and memory. Conventional vector processing requires that scalar operands necessary for execution of vector instructions be available for the scalar processor to issue the vector instructions to the vector processor. Although this prevents memory conflicts between the units, it couples the operation of the scalar processor and the vector processor because the scalar processor does not issue the vector instructions to the vector processor if the required scalar operands are not ready. To decouple operations in a vector/scalar computer while maintaining memory consistency, Applicant teaches and claims the use of the vector processor to predispatch and dispatch vector instructions issued by the scalar processor to the vector processor.

Applicant describes at Fig. 1 and p. 3, lines 14-27, and claims in claims 1-16 a system and method for decoupling operation of a scalar processing unit from that of a vector processing unit in a vector/scalar computer system. For example, Applicant teaches at p. 3, lines 22-23 that the scalar unit (12) can run ahead of vector unit (14), resolving flow and doing address arithmetic. As noted at p. 5, lines 6-7, Applicant also teaches that vector load/store unit (22) executes vector instructions independently from vector execute unit (20).

Applicant also describes at Fig. 7 (block 64) and p. 8, lines 27-28, and claims in claims 1-16, that the vector processing unit includes a vector dispatch unit. For example, Applicant teaches that the vector sector (VS: 52) includes a Vector Dispatch Unit (VDU) (64). *Id.* To run the vector dispatch unit (VDU), as noted at p. 20, line 27 and p. 22, lines 25-27, the Vector Dispatch Logic (VDL) and the Vector Load/Store Dispatch Logic (VLSDL) are used.

As described by Applicant at Fig. 7 and p. 15, line 28, and claimed in claims 1-16, Applicant's decoupling method sends a vector instruction from the scalar processing unit to the vector dispatch unit even if the necessary scalar operands are not ready. For example, Applicant teaches that vector instructions are *dispatched* in-order from the dispatch unit (DU: 82) in the scalar section (50) to the vector dispatch unit (VDU: 64) in the vector section (52). *Id.* Regarding the sending process, Applicant further teaches at Fig. 7 and p. 16, lines 6-7, and claims in claims 1-16, that vector instructions are marked complete by the dispatch unit (DU) at dispatch if they are "not vector memory instructions or require no scalar operands."

As claimed in claims 1-16, Applicant's decoupling method then reads and transfers a scalar operand from the scalar processing unit to the vector processor unit. For example, Applicant teaches at Fig. 7 & 8b and p. 16, lines 10-14 that the dispatch unit (DU) not only dispatches the vector instruction to the A/S unit (90 & 92) but also reads A/S registers (56 & 58) in the scalar section (50), and finally sends the scalar operand(s) to the vector dispatch unit (VDU).

As claimed in claims 1-16, once the scalar operands are sent to the vector dispatch unit (VDU), Applicant's decoupling method predispatches the vector instruction within the vector dispatch unit if the vector instruction is scalar committed. For example, Applicant teaches at Fig. 7 and p. 15, line 29 through p. 16, line 2, that all vector instructions are *vpredispatched* in-order in the vector dispatch unit (VDU: block 64) after all previous instructions are scalar committed.

As claimed in claims 1-16, the predispatched vector instructions are then dispatched if all required operands are ready. For example, Applicant teaches at Fig. 7, p. 16, lines 14-15 & 19-20 and p. 24, lines 7-8 & 11-12 that the VDU (64) *vdispatches* or *vlsdispatches* the vector instruction and data in-order to a vector unit (VU) (66) or a vector load/store unit (VLSU) (68) respectively if all operands are ready.

As claimed in claims 1-16, the dispatched vector instruction is executed as a function of the scalar operand. For example, Applicant teaches at Fig. 7 & 8C and p. 10, lines 5-6 that the vector dispatch unit (VDU) dispatches the vector instructions to the vector unit (VU: 66) and the vector load/store unit (VLSU: 68) for execution. Applicant further teaches, for example, at p. 10, line 25 through p. 11, line 8, that the vector unit (VU) and the vector load/store unit (VLSU) perform the dispatched vector operations.

Applicant also claims in new claims 17-24, that dispatching the predispatched vector instruction includes marking the vector instruction as complete if the vector instruction is not a memory instruction but requires scalar operands and that executing the dispatched vector instruction includes marking the vector instruction as complete after vector address translation if the vector instruction is a memory instruction. As noted at p. 13, lines 11-13 and p. 16, lines 6-7, 14-17 and 22-24, for example, Applicant teaches that vector instructions are marked complete at different time by different units based on whether they are memory instructions and whether they require scalar operands.

§112 Rejection of the Claims

Claims 1-16 were rejected under 35 U.S.C. § 112, first paragraph, as failing to comply with the written description requirement. Specifically, the Examiner states that he could not find support in the specification for the amended and new claimed subject matter and that no indication in the remarks was given on where to locate said claimed subject matter.

Regarding the amended claims 1, 3, 5 and 7-11, Applicant added following three limitations to claims 1, 3, 5 and 7-11 to more clearly emphasize Applicant's claimed invention: (1) (when sending) marking the vector instruction as complete if the vector instruction is not a vector memory instruction and if the vector does not require scalar operands; (2) predispatching the vector instruction within the vector dispatch unit if the vector instruction is scalar committed; and (3) dispatching the predispatched vector instruction if all required operands are ready.

Firstly, as for the marking process, Applicant teaches, for example, at Fig. 7 and page 16, lines 6-7, that vector instructions are marked complete by a dispatch unit (DU: block 82) at dispatch if they are "not vector memory instructions or require no scalar operands."

Secondly, as for the predispatching process, Applicant teaches, for example, at Fig. 7 and page 15, line 29 through page 16, line 2, that "all vector instructions are *vpredispatched* in-order in the vector dispatch unit (VDU: block 64) after all previous instructions are scalar committed."

Lastly, as for the dispatching process, Applicant teaches, for example, at Fig. 7, page 16, lines 14-15 & 19-20 and page 24, lines 7-8 & 11-12, that the VDU (64) *vdispatches* or *vlsdispatches* the vector instruction and data in-order to a vector unit (VU) (66) or a vector load/store unit (VLSU) (68) respectively if all operands are ready.

Regarding claims 12 and 14, Applicant added a new limitation of **decoupling** the storing process and the transferring process as claimed in claims 3 and 4. Applicant teaches, for example, at page 11, lines 4-5 and 9-10, that the VLSU (68)'s memory operation can load vector memory data decoupled from the VU (66) (i.e., vector execution unit including the vector registers) before the data is used in the VU. Applicant also teaches, for example, at page 11, lines 9-10, a large load buffer (LB) (78) is used by a vector section (52) to support decoupling of the VLSU (68) from the VU (66).

Regarding claims 13 and 15, Applicant added a new limitation of "writing memory load data to the load buffer until all previous memory operations completes without fault." Applicant teaches, for example, at Fig. 7 and page 11, lines 12-13, that memory load data is placed in the load buffer (LB) (78) by the VLSU (68) until the VU (64) has determined that no previous memory operations will fault.

Regarding claim 16, Applicant added a new limitation of transferring the data to the vector register by the load buffer if no previous memory operations have failed. Applicant teaches, for example, at page 11, lines 12-14, that the VU (66) moves the load data from the LB (78) into the appropriate vector register (70).

§*102 Rejection of the Claims*

According to *M.P.E.P.* § 2131, which cites *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987), "a claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." In addition, to anticipate a claim, "the identical invention must be shown in as complete detail as in contained in the … claim." *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989).

Claim 1 was rejected under 35 U.S.C. § 102(b) for anticipation by Beard et al. (US 5,430,884, hereinafter "Beard").

Applicant respectfully submits that Beard does not anticipate claim 1 as taught and claimed by Applicant.

Beard describes a method for fetching instructions in an high performance scalar/vector processor. Specifically, Beard describes "issuing" both scalar and vector instructions from an instruction cache in a scalar processing unit to a vector processing unit. Beard further describes "initiating" the issued instructions "normally" or "dependently" based on the availability of the resources required for execution of the instructions. Beard, therefore, fits the description of conventional vector processing systems as described in the Background section of Applicant's specification and as stated in the Summary of claimed subject matter above. That is, systems such as Beard hold the vector instructions in the scalar processing unit until all scalar operands have been calculated by the scalar processing unit.

In contrast, Applicant teaches, and claims in claims 1-24, issuing vector instructions to the vector processing unit even though their scalar operands are not yet known. Instead of checking availability of the scalar operands in the scalar processing unit as in Beard, under Applicant's approach, the vector processing unit checks whether the required scalar operands are ready for execution of vector instructions. *See* below for more detail.

Beard, for example, does not teach following elements as taught by Applicant and claimed in claim 1: (1) marking the vector instruction as complete if the vector instruction is not a vector memory instruction and if the vector does not require scalar operands; (2) predispatching the vector instruction within the vector dispatch unit if the vector instruction is scalar committed; and (3) (inherently) waiting, within the vector dispatch unit, until all operands are ready before dispatching the vector instruction. The Examiner, therefore, has failed to show each and every element as taught by Applicant and claimed in claim 1 as required by *Verdegaal Bros.* and the MPEP let alone completely detailed disclosure requirement by *Richardson* as noted above.

In the Office Action, the Examiner asserts that Beard teaches the three (marking, predispatching and dispatching) elements as described and claimed by Applicant (Office Action, p. 3, # 6, lines 6-15 and p. 24, # 26, lines 5-10). As support of this, the Examiner points to col. 3, lines 21-34 and lines 60-66 of Beard, which states:

> (lines 21-30) Although both vector and scalar instructions are issued from the scalar processor's issue unit to begin the execution process, vector instructions must also "initiate" before they can be executed by an associated vector processor unit.

> (lines 31-36) A vector instruction is decoded and **issued** and transferred [from the scalar processor unit] to a queue in the vector processor unit of the processor when its scalar operand data, if any, is available and the queue is not full. From this queue [in the vector processor unit], the vector instruction is decoded again and "**normally initiated**" for processing if the vector registers and functional unit or data port required by the instruction are available. The vector initiation queue of the preferred embodiment holds up to five vector instructions that have issued but not initiated...
>
> (lines 60-66) The initiation process starts the processing of each vector instruction in the order in which it is received, one instruction per clock (maximum). The vector initiation control mechanism validates the availability of the required vector register resources, and the functional unit or read/write data port. If they are available, the instruction execution is [**normally**] **initiated**. If only the functional unit is available, the initiation logic attempts the **dependent initiation** of the instruction.

Specifically, the Examiner argues that Beard teaches Applicant's two-step dispatching for vector instructions because, under Beard's approach, the instruction is **issued** when scalar data is available (or if it does not require any), and is then **initiated** once the resources for its execution are available. Applicant respectfully disagrees with the Examiner's interpretation of Beard.

Applicant respectfully submits that Beard's 'issue-initiate' approach for executing vector instructions is different from Applicant's 'issue-predispatch-dispatch' approach. First of all, unlike the Examiner's assertion, Beard does not teach marking a vector instruction as complete if the vector instruction is not a memory instruction and if it does not requires scalar operands as taught by Applicant at p. 16, lines 6-7, for example, and claimed in claim 1. Applicant respectfully submits that there is never a reason for marking complete in a system such as Beard as described and claimed by Applicant since vector instructions will never issue until all required scalar operands are present. As noted at Fig. 7 and p. 13, lines 5-6, Applicant teaches that an instruction may graduate if it and all previous instructions in the dispatch unit (DU: 82) in the scalar section (SS: 50) are marked complete and did not trap. The marking process, therefore, is important to maintain memory consistency while decoupling between the scalar section (SS: 50), the vector load/store unit (VLSU: 68) and the vector unit (VU: 66). Applicant is unable to find such a teaching in Beard.

In addition, as quoted above, none of the cited portions show that Beard's vector processor unit 'predispatches' a vector instruction before initiating it for execution. Under Beard's approach, once the scalar processor unit **issues** and transfers the vector instruction to the

vector processor unit, the vector processor unit then **initiates** the vector instruction for execution (normally or dependently based on availability of the functional units and vector resources such as vector read/write data port). In other words, the vector processor unit performs only one step (i.e., 'initiating') before executing the vector instruction issued from the scalar processor unit. This is supported by other portions of Beard (e.g., col. 14, lines 28-59 with emphasis on lines 34-39 and 49-53).

In contrast, under Applicant's approach, the vector section performs two separate steps: 'predispatching' and 'dispatching' the vector instruction issued from the scalar section. Applicant teaches, for example, at page 15, line 28 through page 16, line 20, that the vector dispatch unit (VDU) **predispatches** all vector instructions after the dispatch unit (DU) (82) in the scalar section issues the vector instructions to the vector dispatch unit (VDU). Applicant further teaches that the vector dispatch unit (VDU) then **dispatches** (*vdispatches* or *vlsdispatches*) the vector instructions for execution based on whether all required operands are available as discussed above. These two limitations ('predispatching' and 'dispatching') are claimed by Applicant in claim 1. Therefore, Beard does not teach the vector dispatch unit (VDU) intermediately predispatching the vector instruction before finally dispatching the vector instruction for execution. Furthermore, Beard does not disclose the vector dispatch unit (VDU) using 'scalar committed' status of the vector instruction as a condition for predispatching it. Applicant is unable to find such a teaching in Beard.

Finally, under Beard's approach, as discussed above, the scalar processor unit 'issues' the vector instructions only when its scalar operand data, if any, is available. In order to issue the vector instructions, therefore, the **scalar processor unit** must check the availability of the scalar operands required for execution of the vector instructions **before issuing** them. In contrast, under Applicant's approach, the scalar section (Fig. 7, block 50) does not require that the scalar operands be available to 'send' (i.e., issue) the vector instructions to the vector dispatch unit (64). Instead, the **vector dispatch unit (VDU)** (64) in the **vector section** (52) checks whether the required operands are available **before 'dispatching'** the vector instructions. Applicant teaches, for example, at page 16, lines 13-15 and 18-20, that the vector dispatch unit (VDU) collects all required scalar operands, pairs them with the vector instruction before dispatching the vector instructions to either vector unit (VU) (66) or vector load/store unit (VLSU) (68). As recognized

by Applicant, for example, at page 2, lines 5-11, decoupling or limiting the coupling between the scalar, vector load/store and vector executions units is an important goal of Applicant's claimed invention. The delayed coupling between the vector instruction and its required scalar operands is important to achieve such a goal.

For the reasons discussed above, Beard does not teach a method and system for decoupling operations among the scalar, vector load/store and vector execution units as taught by Applicant and claimed in claim 1. Reconsideration is respectfully requested.

§103 Rejection of the Claims

Claims 2-4, 7 and 12-16 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Beard in view of Patterson et al (hereinafter "Patterson").

Beard is discussed above.

Patterson describes use of a date cache and an instruction cache along with a register, address translation and dealing with a translation fault, and use of a vector load/store unit in a scalar/vector processing computer system.

As noted in the discussion of claim 1 above, claims 2-4, 7 and 12-16 are patentable since none of the cited references, alone or in combination, teach or suggest the following elements as taught by Applicant and claimed in claim 2-4, 7 and 12-16: (1) marking the vector instruction as complete if the vector instruction is not a vector memory instruction and if the vector does not require scalar operands; (2) predispatching the vector instruction within the vector dispatch unit if the vector instruction is scalar committed; and (3) (inherently) waiting, within the vector dispatch unit, until all operands are ready before dispatching the vector instruction.

Claims 5, 6, 8 and 9 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Beard in view of Gharachorloo et al (hereinafter "Gharachorloo").

Beard is discussed above.

Gharachorloo describes using a reservation station and reorder buffers for renaming registers.

As noted in the discussion of claim 1, claims 5, 6, 8 and 9 are patentable since neither Beard nor Charachorloo, alone or in combination, teach or suggest the following elements as

taught by Applicant and claimed in claims 5, 8 and 9: (1) marking the vector instruction as complete if the vector instruction is not a vector memory instruction and if the vector does not require scalar operands; (2) predispatching the vector instruction within the vector dispatch unit if the vector instruction is scalar committed; and (3) (inherently) waiting, within the vector dispatch unit, until all operands are ready before dispatching the vector instruction.

Claims 10 and 11 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Beard and Gharachorloo, further in view of Patterson.

As noted in the discussion of claim 1, claims 10 and 11 are patentable since none of the cited references, alone or in combination, teach or suggest the following elements as taught by Applicant and claimed in claims 10 and 11: (1) marking the vector instruction as complete if the vector instruction is not a vector memory instruction and if the vector does not require scalar operands; (2) predispatching the vector instruction within the vector dispatch unit if the vector instruction is scalar committed; and (3) (inherently) waiting, within the vector dispatch unit, until all operands are ready before dispatching the vector instruction.

Claims 17-24 are added in this amendment. As noted in the summary of claimed subject matter, the limitations in claims 17-24 are fully supported from the Specification (e.g., Spec. p. 13, lines 11-13 and p. 16, lines 6-7, 14-17 & 22-24).

The new claims 17-24 are patentable as being dependent on a patentable base claim. In addition, none of the cited references, alone or in combination, teach or suggest marking the vector instructions complete at different time by different units based on whether they are memory instructions and whether they require scalar operands as taught by Applicant and claimed in claims 17-24. Applicant is unable to find such a teaching in any of the cited references. Allowance of claims 17-24 is respectfully requested.

CONCLUSION

Applicant respectfully submits that the claims are in condition for allowance and notification to that effect is earnestly requested. The Examiner is invited to telephone Applicant's attorney (612) 373-6909 to facilitate prosecution of this application.

If necessary, please charge any additional fees or credit overpayment to Deposit Account No. 19-0743.

Reservation of Rights

In the interest of clarity and brevity, Applicant may not have addressed every assertion made in the Office Action. Applicant's silence regarding any such assertion does not constitute any admission or acquiescence. Applicant reserves all rights not exercised in connection with this response, such as the right to challenge or rebut any tacit or explicit characterization of any reference or of any of the present claims, the right to challenge or rebut any asserted factual or legal basis of any of the rejections, the right to swear behind any cited reference such as provided under 37 C.F.R. § 1.131 or otherwise, or the right to assert co-ownership of any cited reference. Applicant does not admit that any of the cited references or any other references of record are relevant to the present claims, or that they constitute prior art. To the extent that any rejection or assertion is based upon the Examiner's personal knowledge, rather than any objective evidence of record as manifested by a cited prior art reference, Applicant timely objects to such reliance on Official Notice, and reserves all rights to request that the Examiner provide a reference or affidavit in support of such assertion, as required by MPEP § 2144.03. Applicant reserves all rights to pursue any cancelled claims in a subsequent patent application claiming the benefit of

priority of the present patent application, and to request rejoinder of any withdrawn claim, as required by MPEP § 821.04.

Respectfully submitted,

GREGORY J. FAANES ET AL.

By their Representatives,

SCHWEGMAN, LUNDBERG, WOESSNER & KLUTH, P.A.
P.O. Box 2938
Minneapolis, MN 55402
(612) 373-6909

Date January 5, 2007 By Thomas F. Brennan
Thomas F. Brennan
Reg. No. 35,075